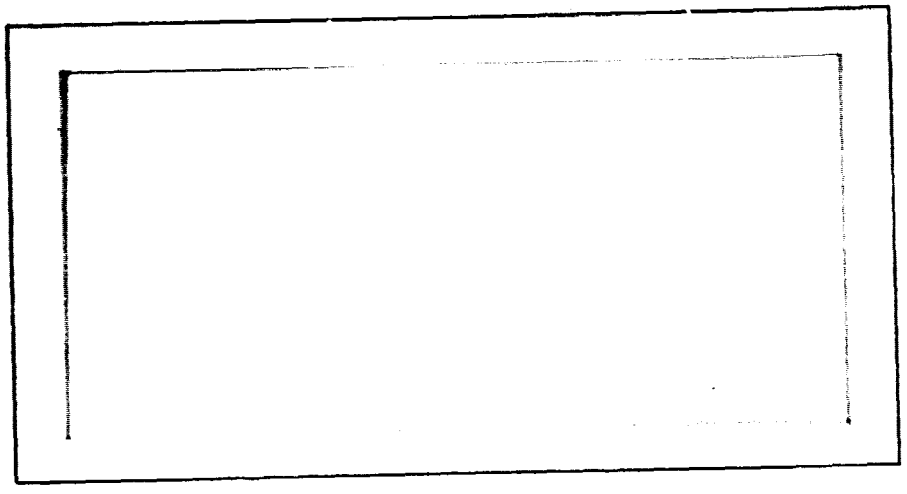
CR 151812



SCIENCE APPLICATIONS INCORPORATED

(NASA-CR-151812) MAN-MACHINE INTERFACE ANALYSIS OF THE FLIGHT DESIGN SYSTEM (Science Applications, Inc.) 37 p
HC A03/MF A01 CSCL 05H

N78-29754

Unclas
G3/54 28501

MAN-MACHINE INTERFACE ANALYSIS OF THE FLIGHT DESIGN SYSTEM

by

H. Rudy Ramsey
Michael E. Atwood
John K. Willoughby

Technical Report No.

SAI-78-089-DEN

30 June 1978

Science Applications, Inc.

40 Denver Technological Center West, 7935 East Prentice Avenue, Englewood, Colorado 80111, 303/773-6900

Other SAI Offices: Albuquerque, Ann Arbor, Arlington, Atlanta, Boston, Chicago, Huntsville, La Jolla, Los Angeles, McLean, Palo Alto, Santa Barbara, Sunnyvale, and Tucson.

FOREWORD

This document is the final report of work performed in the period 1 March 1978 - 30 June 1978 on Contract No. NAS9-15535 (SAI Project No. 1-032-00-129), entitled, "Man-Machine Interface Analysis of the Flight Design System." The purpose of this project was to conduct a brief, broad human factors analysis of the Flight Design System, a system intended for use in shuttle-era flight design by the Mission Planning and Analysis Division, NASA Johnson Space Center. The human factors analysis was intended to provide specific recommendations wherever appropriate, and to identify potential problem areas involving human factors issues.

TABLE OF CONTENTS

INTRODUCTION

Flight design for the Shuttle flights of the Space Transportation System (STS) imposes several new requirements on the Mission Planning and Analysis Division (MPAD) of the NASA Johnson Space Center. The most serious of these requirements is the high flight rate, which is projected to reach approximately 50 flights a year by 1983 and is much greater than that of previous manned spaceflight programs.

Currently, flight planning is accomplished with large, batch computer systems. These systems, however, are not sufficient to support the high flight rate of the Space Shuttle era. Clearly, a new approach to flight design is required. This approach should be a production-oriented system that provides the analyst with automated analytical tools and allows for rapid, interactive flight design. In addition, this system should provide an automated documentation process for the production of standardized flight profiles and associated documents. This would free the analyst from many documentation tasks, which require approximately 50-60% of the analyst's time under the current system, and would allow the analyst to devote more time to actual flight design.

MPAD is currently developing such techniques and aids. The Flight Design System (FDS) is intended to provide a powerful, flexible system for use in flight design. Experience at JSC during previous manned space programs has amply demonstrated that the flight design process requires the judgment and intervention of experienced flight planners. Yet the level of effort required to support anticipated STS flight rates using the flight design techniques of previous programs would be prohibitive. Thus, the FDS must preserve the critical elements of human participation while maximizing the computer's role in the flight design process. This efficient combination of human skills and computer capabilities requires that the human factors aspects of the FDS must be

rigorously evaluated and carefully designed based on established principles of effective man-computer interaction.

EVOLUTION OF THE FLIGHT DESIGN SYSTEM

The Flight Design System concept has evolved from the basic computational techniques already in use in the batch environment of pre-STS flight planning. The functional prototype (FDS-1) which has been developed uses the basic system architecture of the FDS concept, but contains application processors derived, for the most part, directly from existing batch tools, especially in the trajectory area.

It is important to recognize that the purpose of FDS-1 is to allow testing of the basic system architecture and functions. The prototype system provides a good, general-purpose capability, but possesses only a limited set of computational aids. Furthermore, up to the present time, little or no explicit attention has been given to human factors issues. This was a conscious decision on the part of MPAD. It was generally felt that resources should be concentrated on achievement of a working basic system capable of providing an experimental testbed, and only then should human factors consultation be sought. Because of the very flexible nature of the FDS architecture, modification of fairly significant functional aspects of FDS is still possible prior to completion of FDS-2, the first production version of the system. It is in this context that the human factors analysis reported here was conducted.

NATURE AND OBJECTIVES OF THE HUMAN FACTORS ANALYSIS

Historically, the field of human factors has dealt heavily with such areas as controls and displays, psychomotor tasks, and the application of task analysis and similar job-analytic techniques. Even today, many people associate the phrase "human factors" with activities of this sort. While these activities are still relevant, the last decade has seen a much greater emphasis on cognitive tasks, problem-solving aids, and analysis of the problem-solving behavior of computer system users. This has resulted in part from the increasing maturity of the study of cognitive psychology and human information processing, and in part from the rapidly increasing use of computers as aids for problem-solving tasks, whereas earlier computers were used primarily to support clerical tasks.

As a result of this shifting emphasis, human factors personnel have begun to make significant contributions not only in the area of "knobs and dials" -- the design of keyboards, formatting of displays, etc. -- but also in the much more basic and significant areas concerned with analysis of user information requirements, basic functional design of the system, detailed dialogue design, and even the overall problem-solving procedures of the user, of which the interactive tool is only a part.

The objective of the current effort was to perform a broad analysis of the human factors issues involved in the design of the Flight Design System. The analysis was intended to include characteristics of the system itself, such as:

- The basic structure and functional capabilities of FDS
- User backgrounds, capabilities, and possible modes of use
- FDS interactive dialogue, problem-solving aids
- System data management capabilities

and to include, as well, such system-related matters as:

- Flight design team structure
- Roles of technicians
- User training
- Methods of evaluating system performance

From the start, it was understood that the small size of this effort would prevent the development of detailed recommendations in many of these areas, but it was felt that a rapid, broad identification of the issues would be the most cost-effective use of the available resources. Wherever possible, specific recommendations have been made. In other cases, we have identified the issues which seem most important and, in some cases, have suggested additional analyses or experiments which might provide resolution.

SOME BASIC ISSUES

There are several basic issues, pertaining to the design and use of FDS, that are particularly important with respect to a human factors analysis. For convenience, we have organized these issues into six basic categories. As will be seen, however, these categories are not completely independent and consideration of an issue in one category may well have implications for issues in other categories.

1. STS planning requirements can be satisfied only through a new approach to the problem of flight design. In previous manned spaceflight programs, planning was partitioned by mission phase (ascent, on-orbit, deorbit) and each phase was considered individually. Such an approach requires effective interfaces among the personnel working on each phase. Such interfaces impose communication and sequencing problems and may tend to produce "bottlenecks" in the flight design process. Given the high flight rate projected for the Space Shuttle era, such bottlenecks must be avoided.

Under the FDS concept, planning tasks will be partitioned primarily by expected planning difficulty rather than by mission phase. This requires flight design analysts to employ a new approach to flight design. A related issue is the use of a team concept. As planned, a flight design team will consist of 6-9 personnel, possibly from different disciplines, and will include a team leader who will function as data base manager and have primary responsibility for approving flight designs. Less complex flights may be designed by teams of only one or two members. This concept may require more user versatility than previous approaches to flight design. In addition, the allocation of tasks to team members, documentation, communication paths, etc. affect, and are affected by, the structure, or organization, of the team. Some structures would likely be more effective than others. These issues are

not only important in their own right, but have strong implications for the basic properties and detailed design of FDS.

MPAD has no experience with the planning mode which will be required. If experience and performance data are required to validate, or to develop further, the FDS concept, explicit experimentation will be necessary.

2. The users of FDS will have highly varied backgrounds. Users will range from highly skilled engineers to technicians. Even among the skilled engineers, there will be differences with respect to familiarity with interactive systems and there may even be differences in the problem-solving approaches that are applied to flight design. It is difficult to assess the impact of technicians since the training, experience, and abilities of technicians are not yet well known.

How technicians should be trained and used is a particularly important issue. This issue has implications for the overall planning approach, the organization of flight design teams, and the design of FDS. It is important to consider not only the abilities and training of technicians, but also the acceptance of the technician role by engineers.

The training of technicians is of primary importance. Technicians must be taught the fundamentals of flight design, but the associated physics, mathematics, etc., must be highly simplified. In effect, technicians must be presented with an abstract view of flight design. Care must be taken to ensure that all relevant aspects are included in this abstraction and only inessential aspects are excluded.

3. Satisfactory planning results, in a high flight rate environment, will depend on the development of appropriate planning interfaces with

related planning tasks. These tasks include utilization planning, crew activities planning, flight simulation, etc. Developing effective interfaces may have implications for the design of FDS.

Frequently, the conversion to interactive aids, shorter span-time, and less labor-intensive planning -- beneficial though the transition may be -- is accompanied by a more iterative approach to planning. This may be because iterations are easier to accomplish, or simply because errors may easily persist into a later part of the planning cycle before detection. Whatever the cause, this phenomenon can spell disaster if the interfaces among the various stages of the planning cycle are cumbersome.

4. The current FDS concept relies heavily on an assumed independence among the analytical steps, which are applied in a sequential, linear manner. Such linear planning may be, in some cases, incompatible with the normal problem-solving behavior of the users. In many kinds of planning tasks, problem solving proceeds in a hierarchic, rather than linear, manner. Such considerations have implications for the design of the FDS dialogue and also for the interfaces between application processors.

5. The success of FDS may well depend on the degree to which it aids the analyst in retrieving and recognizing problem-relevant information. Such retrieval and recognition often relies on episodic memory and other cues that are difficult to reproduce or replace in an automated system. Examples of episodic memory are recalling that the document you need is "the one with coffee stains" or that the necessary formulas are "in the book with the green cover." Other relevant cues include abstract labelling. The information, by itself is not recognized as important; rather, the problem solver recognizes the abstract label, or mark associated with that information. The success of FDS may depend

development and use of appropriate aids for data labelling, abstractions, retrieval, etc.

6. Satisfactory performance of planning tasks involving FDS will depend heavily on the nature of the relationship between the user and the system. Of primary importance is the issue of user acceptance. In addition to the usual aspects, this may also involve overcoming any perceived threat (loss of prestige, job security, etc.) on the part of flight design personnel. This is best addressed by an appropriate transition from current practices to training program to operational use of FDS.

The success of FDS also depends on an appropriate match between FDS functional capabilities, dialogue, etc. and the background and experiences of FDS users. For example, if different users have basically different approaches to problem solving, require different types of dialogue support, etc., FDS must accommodate these differences.

The functional capabilities and dialogue of FDS must also match the possibly diverse requirements of the various problem-solving tasks involved in planning. For example, if ascent, time-line scheduling, reentry, etc., are perceived by users as being different types of problem-solving tasks, this has implications for the design of FDS.

FINDINGS

As currently planned, FDS is potentially very effective as a basic computational aid. Its general-purpose structure is quite powerful and flexible. Although we have a number of recommendations for changes in the dialogue, and other properties of the system, very few of them are in any way incompatible with the current basic structure. It would appear, however, that the currently-planned FDS system may be difficult for computer-naive users to operate. This situation appears to be significantly improvable by the incorporation of some relatively inexpensive dialogue improvements, as suggested below. Wherever possible, we have tried to provide those solutions which appear to give the "most bang for the buck," since they have the greatest probability of being accepted and incorporated.

In addition to simple dialogue changes, more basic improvements, such as problem-solving aids and tutors, appear quite promising. However, the small current effort did not permit a sufficiently detailed analysis to develop them in any detail. In most cases, the design of such aids requires a more detailed understanding of the users' problem-solving practices than now exists for the flight design community, and further, study of that problem-solving behavior appears to be the most effective next step. In any event, such aids are unlikely to conflict with the basic design of the system.

Understanding the Analytical Process

Understanding the problem-solving behavior of the users is basic to the formulation and evaluation of appropriate design aids. We were able to conduct a number of interviews of flight designers in several specialty areas (ascent, trajectories in general, attitudes, consumables). The designers were asked to describe flight design problems and procedures, tools (including even hand calculators), experience with interactive systems as well as batch systems, and their experience, if any, with FDS-1 in particular. The "critical incident technique" was also employed in an informal way. In this method, personnel are asked to identify instances of outstanding success or failure (we addressed only the latter) of the process (flight design) in which they are involved. Inquiries of this sort often help to identify particularly weak elements of the process which might be improved through automated aids, better procedures, etc. Most of the interviewed designers responded quite readily to such questions, and some of the responses have a direct bearing on FDS design. For example, a Gemini problem was identified which appeared to be due to a failure to update early planning data as more exact information became available. An automated aid, such as FDS, might assist with such a problem by means of data dating and automatic flagging of old data.

While these techniques were informative, they are entirely inadequate for the formation of a detailed understanding of the flight design process. Particularly in an abstract area like flight design, few people are able to describe in detail the way in which they solve problems, even though they may be quite expert at solving them. Furthermore, actual practices and performance often differ from the individual's perception of what he does. To achieve more exact knowledge of flight design behavior, it will be necessary to conduct simulations and observation of actual flight planning behavior. We believe that the

benefits of such simulations would far outweigh their costs, particularly if they are conducted early enough to impact the tutorial and problem-solving aids incorporated in FDS-2. One of the major recommendations of the current study is that MPAD conduct such simulations.

At first glance, the use of FDS-1 as a computational aid in such simulations seems quite reasonable. However, it should be noted that FDS-1 lacks a comprehensive set of computational aids, and concentrates primarily in the trajectory area. A more basic difficulty is the fact that FDS does not support the early, abstract portion of the planning process at all (see later section on "Abstract Plans"). This may, or may not, be a deficiency in terms of automated support of the flight design process, but is does make FDS a potentially inappropriate tool for gathering data about the whole process. It would appear preferable to conduct at least the early simulations using current manual (batch-aided) planning methods, and extend the simulations to involve FDS only when that step is clearly justified.

Another type of data-gathering effort is also desirable. Responses to the critical-incident quesions used in the interviews indicate that this may be a source of useful information. Applied in the form of a survey, rather than face-to-face interviews, this approach could be inexpensively applied to the entire user community. With an appropriately designed questionnaire, this could be a very cost-effective source of information.

Technician Users

The role in which technicians will be used in STS-era flight design is not yet clear, and may vary from a simple clerical aide to complete end-to-end planning of simple missions, under engineering supervision. Appropriate FDS properties, as well as design procedures, depend somewhat on this determination, which depends in turn on a determination of technician capabilities, acceptance of technicians by engineers, etc.

At the very least, it is clear that technician users will require a more tutorial dialogue than will the experienced engineer. It is important to recognize that the technician is new to both flight design and the use of interactive computer aids. Thus, the FDS dialogue must not only be sufficiently tutorial to allow mechanical operation of the system by relatively inexperienced system users -- a provision appropriate even for the engineer users, many of whom are experienced only with batch tools -- but must also provide flight design information in a form comprehensible to users who are relatively inexperienced in the entire area of manned space flight. A system which provides extensive guidance with respect to interactive dialogue, but little assistance in understanding the meaning of maneuvers, flight planning operations, etc., might be usable to engineer users but inadequate for technicians.

More extensive problem-solving aids might be particularly effective for the technician user, who will presumably be performing most of his work in accordance with planning procedures which will be known and fairly well structured. The difficulty is that they are not presently known and well structured. A basic issue here is the selection of an appropriate problem abstraction for use by the technician. Clearly, it will be necessary to provide the technician with a more

simplified view of the problem than that possessed by the engineer, and yet avoid omission of critical elements. For example, the use of a graphical analogue, in which the technician relies primarily on graphical portrayals of trajectory and attitude information, is under consideration. While this may very well prove to be an effective approach, its implementation requires more knowledge of the technician user class than we now have.

MPAD has proposed to conduct an experimental investigation of the use of technicians in flight design. This study would involve the formulation and trial of various approaches to problem abstraction, training, etc., with a small number of technicians. We believe that this study is a very important source of information which will impact not only FDS design (at least as regards tutorial and problem-solving aids), but also flight design procedures, team structure, etc. We would encourage MPAD to proceed with such a study as soon as possible.

Abstract Plans

In many kinds of planning behavior, the planner begins with the development of an abstract plan which is at a relatively high level (e.g., "launch, then orbital entry, then ...") and contains little, if any, of the detail found in the final concrete plan ("..., then perform 3 fps delta-V maneuver, then ..."). Development from the most abstract form may be quite direct, or may involve many steps, each of which represents a slightly more concrete statement of one of the elements of the hierarchic plan. In flight design, for example, an intermediate planning step might be "perform maneuvers to attain geosynchronous orbit," which is subsequently broken down into a series of specific maneuvers.

It is important to recognize that FDS currently does not allow the formulation, storage, or refinement of an abstract plan. The computations with which FDS is concerned are the most concrete elements of the hierarchic plan. The relationships of these concrete elements to the abstract plan are not explicitly represented in FDS. Furthermore, it is possible for FDS application processors to cross stage boundaries, further complicating the recognition of these relationships.

The absence of abstract plan information in FDS is probably not a problem for the experienced engineer. In the case of this user, the concrete plan is a familiar representation, and its ties to abstract plans are "overlearned" and obvious. In the case of the technician, or even a new engineer with no specific flight design experience, this may be a much greater problem. This user is already overloaded with new information, and the meaning of "GPMP DELV10," not to mention the relationship of this specific computation to the overall flight plan, is likely to be less than obvious. Furthermore, experience with planning processes of this sort would suggest that the technician user may be able to perform satisfactorily in a "schema plus corrections" mode -- in which an existing basic plan is selected and modified only in those details necessary to make it applicable to a specific mission -- but that success with this planning mode requires comprehension of the entire plan and all its elements, abstract and concrete.

At present, we do not have sufficient data concerning the problem-solving behavior of either experienced engineers or technicians to determine whether -- or how -- abstract planning information should be incorporated into FDS. At a very specific level, though, it appears likely that comments placed directly in the sequence table might assist considerably in those instances in which the user must comprehend the function of the table. Comments and related features will be discussed in more detail in a later section.

Some General Comments on FDS Dialogue

It is in the area of interactive dialogue that the current study has produced its most specific recommendations. The next few sections of this report deal with various aspects of FDS dialogue, but will be prefaced with a few general comments which do not clearly fit into the later, more specific discussions.

First, the existing (FDS-1) system requires the transmission, from the terminal, of a blank followed by a carriage return to identify a null input. The requirement for a blank character is an extremely error-prone feature. Users generally consider the blank to have no significance, and it is not displayed. Even though modification of the 21MX terminal handler may be required, it is our recommendation that this problem be corrected so that a simple carriage return is consistently recognized as a null input.

In general, a null input should cause only nondestructuve actions, and should usually cause that action which is the most probable need of the user (subject to the constraint that the overall pattern of default actions must be clear and consistent). There are a few instances in the current dialogue in which improvements in default behavior appear possible. We would suggest that a link analysis of the dialogue is the best basis for decisions concerning the dialogue default structure. A link analysis is a simple tabulation of the frequency with which users select each option available at each decision point in the dialogue. Such an analysis is relatively inexpensive and can result in noticeable improvements in dialogue usability, especially for experienced users.

In general, destructive actions (such as deletion of the user's temporary files) should require at least one explicit user action, and preferably two actions, one of which is explicit. In the current FDS

design, a user who has built a temporary file with an editor may type "%" to return to the FDS executive module. Although no explicit permission has been given by the user, this action results in immediate deletion of the file. In other cases, "%" is a normal, nondestructive method of returning to the executive. Unexpected destruction of files can be costly and extremely irritating to the user. We would suggest that, whenever such implicit file deletion is about to occur, an explicit inquiry be made of the user (e.g., "FILE HAS BEEN MODIFIED. OK TO DELETE?"), and that an affirmative response be required ("Y" or "YES") before such deletion proceeds. The cost in extra terminal operations is small and infrequent, but the avoidance of an inadvertent deletion of an extensively modified file can be a significant savings. Of course, this procedure should be used only if there is a modified file which is about to be lost.

Interface of Application Processor to Interface Table

In the current FDS design, resolution of all interface table values must be accomplished before the corresponding application processor is called. This has resulted in the frequent adoption of an array format for those interface tables for which a variety of variables may be required depending on the option selected (e.g., the General-Purpose Maneuver Processor). This is an undesirable practice, since the array name is not an adequate prompt for the required variable(s), and extended prompts are precluded by this practice. It would appear that the most appropriate solution to this problem is a change in the "binding time" of interface table values, so that interface table variables are only required at the time they are requested by the application processor. This would allow the use of explicit variable names, and extended prompts, in the interface table, without necessitating that values be provided for variables which are irrelevant for the processor option(s) selected.

The current FDS design does not allow the user a convenient mechanism for ascertaining the relationships between application processors and interface tables. It would appear desirable that abstracts be provided for both of these elements, and that the user be able to ascertain the answers to both of the following questions: "What application processor is this interface table associated with?" and "What interface tables do I have which are associated with this application processor?"

Application Processor Standards

The usability of a system as complex as FDS is strongly affected by the consistency of the conventions used throughout the system. At present, most application processors operate in a manner similar to batch programs, with the interactive aspects of the dialogue controlled almost entirely by the FDS executive and execution processors. As the conversion to FDS occurs, however, it is evitable that more interaction with the user will occur at the application processor level. This is, in fact, the best way to achieve a highly interactive dialogue, where it is needed, within the confines of FDS. At present, no standards exist which will control the nature or consistency of these interactive aspects of application processors. We recommend that such standards be established, to include at least the following aspects of application processor function: (1) informative comments, (2) error messages, (3) basic interactive dialogue conventions, and (4) disposition of intermediate results. The latter concern stems from an instance observed in FDS-1, in which an intermediate result was displayed on the terminal, but not otherwise saved, even though it was needed for later processing.

Movement Among Subsystems

FDS has several system-level modules (executive, execution processor, editors, etc.). These modules are related to one another by means of the system's access structure. That structure is strictly hierarchic, with the executive controlling access to all other modules. Thus, a user who wishes to access the interface table editor from the execution processor must first return to the executive. It is important that such a system access structure satisfy two criteria. First, it must be sufficiently simple and natural that the user can form a usable "mental model" of the system. The current FDS structure appears entirely satisfactory on the basis of this criterion. Second, it must be functionally adequate and usable from the viewpoint of the specific tasks which the user must perform.

While the FDS structure is functionally adequate, it may be inconvenient in some situations. In particular, the user who is executing a sequence table in the semi-automatic mode cannot conveniently invoke the interface table editor to preview a table. If data are missing from the interface table, the interface table editor is invoked, as if it were a subroutine, with automatic return to the calling environment upon completion. To accomplish the same function for the purpose of a preview, the user must perform four separate access operations (executive, editor, executive, execution processor) and must recreate the original environment by providing a sequence table line number. MPAD should perhaps consider the adoption of a call-and-return approach to editor invocations in general.

The use of single, special characters to change system modules, and as prompt characters for the modules, involves a tradeoff. Such abbreviation can be very helpful to the highly experienced user, but is usually undesirable because its arbitrary conventions are difficult

to learn. In the present case, however, the system structure is simple and the use of such special characters appears to us to be acceptable. Rather than eliminating them, we would suggest the provision of a "wordy" mode (see later discussion of "Dialogue Modes") as an appropriate antidote to the difficulty which the new user has with such a scheme.

Command Language

For those few commands which involve the specification of more than one parameter (e.g., "SEQEDIT TABLEX, TABLEY"), FDS uses a positional notation in which the significance of a particular parameter value is derived from its position in the parameter list. Positional notations are very natural from the viewpoint of the computer processing required in their interpretation, but constitute a significant source of transposition errors by the user. (Which is the old table in the above command, and which the new?) Positional notation is minimal in the current FDS design, and is not particularly objectionable in those specific instances in which it occurs. It is suggested, however, that any tendency for such notation to proliferate, as the system command language grows, should be carefully considered or avoided altogether.

It is desirable that the particular verbs used in any command language be easily discriminable. Ideally, they should be discriminable both alphabetically (so that commands can be specified by the experienced user with the fewest possible characters) and semantically (in the sense that the user readily associates each command with the function which it performs, rather than confusing two or more commands). The FDS command set would benefit from an analysis intended to increase discriminability of commands. As an example, the commands "RESTORE" and "RECALL" appear to violate both criteria. Complete elimination of

such confusability is difficult, and may prove impossible to attain, but considerable improvement should be achievable over the current command set.

When commands have been made maximally discriminable in an alphabetic sense, it is possible to provide the experienced user with a highly abbreviated mechanism for command input. An appropriate mechanism might, for example, be a multiword, first-k-character matching algorithm for command recognition. Such an algorithm might allow

SEQUENCE EDITOR

to be activated with any of the following abbreviated commands

S
SE
SEQ
SED
SEQEDIT
SEQUENCE

With appropriate use of space and comma terminators, it may also be possible to stack commands.

Data Management

It is clear that the recognition, retrieval, and purging of data is potentially a major problem. Two simple mechanisms seem to hold promise for helping with very limited aspects of this problem. First, data dating, along with automated mechanisms for the detection of out-of-date data, may help to prevent inadvertent use of preliminary planning data for final mission planning. Second, automated generation of abstracts for data elements (see next section) may help to insure that these elements are later identifiable. More basic solutions are undoubtedly needed, but will require more analysis, and more experience with FDS, than has been accumulated at present.

Comments and Automatic Abstracting

For several reasons, already identified in earlier sections, the provision of sequence table comments appears desirable. Such comments might serve at least three purposes. First, if used appropriately, they might greatly improve the comprehensibility of the sequence table itself. They might indicate the mission for which the sequence table was built, indicate basic mission phases (the abstract plan) and indicate more clearly the function of individual application processor invocations. These comments should perhaps be of explicitly different types, with an indentation scheme used to discriminate comments associated with single sequence table entries from those associated with mission phases.

A second use for sequence table comments is in the generation of appropriate user feedback during sequence table execution in automatic and semiautomatic modes. Combined with application-processor-generated comments, these comments might provide considerable information to the user if an appropriate "wordy" mode is selected. As the system is currently configured, the automatic mode user may be asked to supply a parameter value for, let us say, the General-Purpose Maneuver Processor without any information about which of a series of maneuvers is involved.

A third use for sequence table comments is in the automated generation of abstracts for data elements. Since these files are system-generated, they will be identifiable only if the system or the user explicitly adds identifying information. An automatically generated abstract might contain such information as the following:

FLIGHT (taken from flight comment in sequence table)
DATE (provided by system)
USER (provided by system)

MISSION PHASE (taken from mission phase comment in sequence table)

SPECIFIC MISSION STATE (e.g., "STATE AFTER 10 FPS DELTA-V MANEUVER" -- obtained by combination of "STATE AFTER" with specific application-processor-execution comment in sequence table).

This example also illustrates the fact that the system may make specialized use of multiple, explicit comment types in the sequence table.

Editor Function

Several specific recommendations appear warranted with respect to editor functions. The default mode of operation for the interface table editor is a "fill in the blanks" mode in which the editor automatically jumps to the first unresolved variable in the table and prompts the user for its value. This is probably the appropriate default mode for automatic execution of this editor to obtain missing values during sequence table execution. It probably is not the appropriate mode if the user overtly requests the editor, perhaps for the purpose of looking over the table. In fact, the existing scheme is highly error-prone. The user who enters the editor and attempts to list the table is likely to find (or even not realize) that he has typed the value "LIST" for a previously unresolved variable. A similar result is obtained by attempting to escape the "fill in the blanks" mode by a null input. The blank symbol currently required to indicate a null input is entered as the variable value. These difficulties may be resolvable by use of a normal editor default (pointer at top of file, prepared to accept list command or explicit variable value substitution). It should be noted that an editor pointer, with the

capability to list only a few lines of the table, may be required by the larger interface tables which will result from the use of explicit variable values. In the "fill in the blanks" mode, a blank input should probably be disallowed unless quotation marks are used, and a similar treatment of "LIST" may be warranted.

A very common sequence of user actions involves replacement of a variable value followed immediately by the printing of the same line. The provision of an automatic "Verify" mode, in which altered lines are immediately redisplayed, would not only reduce the number of user actions required, but might also increase the probability of detection of erroneous entries.

As discussed in a previous section, file deletion on exit from the editor should probably require an explicit user action.

Error Messages

In general, any error message should implicitly or explicitly convey to the user the following information:

Source of Message (Executive, application processor, etc.)
Nature of Error
Severity of Error and/or System Action Taken
User Action Required to Correct the Error

Thus, a reasonable message might say

GPMP: NEGATIVE START TIME WAS SPECIFIED. GPMP TERMINATED.

This message explicitly contains the first three categories of information, and specifies the nature of the error in sufficient detail to clearly indicate the required corrective action. Few, if any, existing FDS error messagessatisfy these criteria, and some messages fail to satisfy any of the four. The adoption of appropriate error message standards, based on the above criteria, is recommended.

A second concern with the existing error messages is the presence of unnecessary encoded information in the text of the messages (e.g., "XCO2"). While such information may be ignored by the experienced user, it can be quite confusing to the novice. If such information is needed by system developers, it should be made available via a "Debug" mode, or an extended message replay capability.

Dialogue Modes

Several dialogue mode possibilities should be considered. Dialogue modes allow a one-time-per-session or even one-time-per-user selection of options which continue to control the dialogue thereafter until the user's election is changed.

A "wordy" mode appears to be strongly desirable, and might even involve multiple levels. When the wordy mode is selected, the user is given extensive feedback concerning system function including comments generated from sequence tables and by application processors, automatic use of extended prompts, etc. This mode is intended primarily for use by novice system users, but may be of continuing utility to technician users.

Several tutorial modes are possible, including:

A general computer-initiated mode in which all user inputs are responses to specific queries by the system.

An argument query mode in which explicit prompts are made for the value of each parameter required or allowed with a command (the user must first type the command).

An abbreviate mode in which the system shows the user the most abbreviated way in which he might have specified the command he has input. Most systems fail to make any provision for teaching the novice user the "tricks" available for rapid use.

An expand-and-verify mode in which the system responds to each command with a fully expanded statement of the command, including default values, and awaits user verification that the command represents the desired action. This mode provides a painless way for the novice to try out his expert-user skills.

A "Debug" mode is the appropriate mechanism for obtaining extended error messages, trace information, etc., which is of use only to the system developer.

The use of an editor "Verify" mode has already been discussed.

User mode profiles, which allow semipermanent selection of all desired modes by the user, are a considerable convenience in systems with widely varying user preferences or experience. Such profiles might be made hierarchic, so that the user may independently select modes or may simply specify the "Beginner" pattern, for example.

The tutorial information provided by the system should not only be made more extensive, but should probably also be made more context-specific.

Use of Storage-Tube Terminal

The selection of a storage-tube terminal for FDS imposes severe constraints on the interactive dialogue which the system can support. Computer-initiated command construction (as by menu-selection) is too slow to provide a viable option, and displayed information must be accumulated, rather than replaced, whenever update rates are faster than a few per minute. These constraints may force revision of the existing application processor display philosophy to reduce the amount

of information routinely displayed.

Determination of the specific way in which this terminal should be used (dialogue, display formatting, etc.) requires a detailed analysis which is beyond the scope of the current effort. A few specific suggestions can be made, however. First, it will be necessary to have the system keep track of the state of each terminal, in the sense of the amount of information currently displayed. The system should stop transmitting to the terminal when its display is full, and await a "go ahead" command from the user. It should then issue a page clear command, redisplay the most recent information, and continue. Although the user can manually clear the display, any information transmitted during that operation (which takes 1-2 seconds) is lost.

Display windowing, in which the display is divided into several distinct areas, is probably the most appropriate general approach to display formatting in this situation. In particular, it is suggested that the "run log" information -- in which successive lines typically represent successive computational steps -- be physically separated from the application processor displays, which are typically multiline displays. This should reduce confusion of "temporal" and "nontemporal" cues.

Finally, the possibility of locally buffered write-through displays to enhance the dialogue, should be considered. This feature should not be purchased without a detailed understanding of its intended use, however, as it alleviates only a few of the constraints outlined above.

"Audit" Trail

It is highly desirable that the system maintain a record of user-system interaction at each session. This record should be replayable, and thus should be sufficiently detailed to permit a complete regeneration of the session. With the simplifying assumption that the system need not be concerned with the initial state of permanent data bases, complete regeneration of the session requires only a record of the user's inputs, which are not ordinarily voluminous. On replay, the system should allow mode changes (as in turning on "Debug"), single-step execution, restart from an intermediate point, etc.

There are several reasons for our belief that this feature is needed. The most basic involves improved communication between the technician and the supervising engineer. In the event that the technician attempts to carry out the engineer's instructions, and the result is unsatisfactory, the fault may lie with the instructions, the technician's execution of the instructions, or the system. In many cases, literal regeneration of the session by the technician will be impossible. If the difficulty is to be assessed and corrected with minimum thrashing, session regeneration by the system is the indicated mechanism. The engineer can simply step through the session with the technician, hopefully recognizing and correcting the error.

A system replay capability is also needed to assist in evaluating user complaints. Often, the user encounters an apparent system malfunction, but cannot regenerate the error. Often, it is impossible for system developers to even determine whether a system or user error is involved, and actual isolation and elimination of the system error usually requires that it occur in the presence of system personnel. The ability to replay the session, especially with trace and extended error message facilities turned on, can be quite advantageous. The replay capability can also be useful in debugging user programs.

Finally, the replay capability can be of immediate use to the user who has a "Where am I?" problem, or who needs to return to a previous computational state in which no "checkpoint" or "save for later use" operation was performed.

Problem-Solving Aids

In addition to dialogue aids (tutorial features, etc.), which assist the user with the mechanical operation of the system, FDS might be made to provide guidance with respect to the comprehension or solution of the planning problem itself. These aids may range from very simple aids (e.g., the use of comments in sequence tables, as already discussed) to very complex and powerful decision aids.

Aside from comments, another simple mechanism which might assist with the representation of abstract plans in the system is the appropriate use of hierarchic sequence table structure, in which one sequence table is allowed to execute another. This feature might allow an overall mission sequence table to refer to separate tables for mission phases, etc. This capability appears especially desirable if greater provision is made for utility functions (initialization, etc.) in FDS-2, as appears to be planned. It is desirable to avoid a dominance of the sequence table by commands which are not logically important to the understanding of the flight design.

If technician users are found to be sufficiently capable that they become actively involved in sequence table construction, aids for this task will probably be needed, and such aids may be useful even for experienced engineers. A simple approach to such aids involves construction of the sequence table by menu selection (which type of launch do you want, which type of entry, maneuvering, etc.?)

More powerful problem-solving aids are undoubtedly possible, and may be desirable, but their design should be based on the results of the recommended simulation study, the technician study, and more extensive experience with the use of FDS-1. Tutorial dialogues could be useful both in aiding the skilled analyst in the use of FDS and in aiding the less skilled analyst in the flight design process itself. A related issue is the use of "critics" or "pre-condition checkers." Precondition checkers are used to ensure that the appropriate preconditions for the action the user is currently initiating are satisfied. These preconditions may include data structures, interface tables, previous actions that are required, etc. Critics are special purpose aids that are intended to either correct or bring to the user's attention errors that frequently occur in the flight design process.

The users of FDS will differ with respect to abilities and previous experience. While the highly skilled user may be expected to recognize the need for help, either in the use of FDS or in the flight design process, this should not be expected of the less skilled (especially the technician) user. Techniques that monitor user actions, recognize when the user is in trouble, and offer appropriate help could be useful. This may, in part, be accomplished by providing for an "audit trail" that records user analytical actions and responses (as discussed in a previous section). An audit trail would also be useful for determining what information users want to retain from one iteration of a problem to the next and, independently of problem-solving aids, would provide information essential to evaluating the use of FDS.

Subsystem Y Interface

It may be necessary to make the subsystem X - subsystem Y interface bidirectional, in the sense that data can be passed in either direction and used for normal computation in the receiving system. This would allow

iterative solutions involving both systems, which may or may not be desirable or necessary. It has not yet been demonstrated conclusively that such iteration is unnecessary. At a more mundane level, though, the ability to pass data from subsystem Y to subsystem X is probably required to support the automated production of planning documents. To the degree that merging of data from both systems into a single page is required, the present concept requires that the merging be performed by FDS (subsystem X). Failure to provide a bidirectional interface would require this operation to be done manually.

Transition to FDS Planning

If MPAD is to fully realize the advantage gained through implementation of the FDS-1 prototype system, it is necessary that FDS-1 be exercised broadly and formally. An approach which merely makes FDS-1 available to those who are inclined to play with it is probably inadequate, since the personnel who will elect to try it out are not representative of the eventual user population. There are both advantages and disadvantages to an early involvement of those users least experienced with interactive aids. Obviously such involvement can result in the recognition of system deficiencies which might not otherwise be discovered early enough for correction in the normal FDS-2 development effort. On the other hand, FDS-1 has intentionally omitted many of the features(tutorial dialogues, etc.) which might make the system easy for these users to learn and operate. Probably a small number of such users should be carefully briefed and invited to use the system. In any event, it is highly desirable that the evolution toward use of FDS for real production planning begin now, in those areas in which FDS-1 provides adequate computational support.

Evaluation of System Performance

A proper approach to the evaluation of FDS-aided planning performance requires the development of: (1) appropriate problem scenarios, (2) good performance criteria, and (3) a performance baseline, probably based on manual solution of the same problems. The simulation study already recommended for the detailed study of flight designer problem-solving behavior would involve development of these materials and data. It should, therefore, provide an appropriate basis for performance-based evaluations of FDS. It is also appropriate to obtain subjective evaluations by the users, using appropriate psychological scaling techniques.

Flight Design Team Structure

This is a very complex issue which was not explicitly made a part of the statement of work performed here. It must be recognized, however, that the structure and role allocation associated with the use of flight design teams have implications for the design of the system itself. For example, the use of a team approach implies that a given individual may be working on several missions in parallel. This in turn suggests the possibility that automated planning status displays might be a useful adjunct to FDS planning. The use of large teams for relatively difficult missions may allow engineers to continue to work in a fairly compartmentalized way, while small teams require an ability to use automated planning tools outside the user's accustomed area. This has clear implications for the system's tutorial and performance feedback features, application processor interfaces, etc.

Clearly, decisions concerning team structure should be based on an understanding of the planning methods employed by flight designers, and of the role of technicians in STS flight design, both of which are the subjects of possible future studies. These decisions should be deferred,

then, but it is also important for MPAD to begin obtaining experience with the use of FDS-1 for team problem-solving. This may be a source of useful insights with respect to both team structure and FDS design.

Automated Production of Planning Documents

The prototype approach which has been developed looks basically very good. Two specific features may warrant consideration. First, the placement of variable names or labels directly in the master text, rather than the use of serial association, would reduce the maintenance effort necessary when the text is modified, and might make the master text more readable. On the other hand, this feature presents some difficulty with arrays and inverted arrays, and renders the spacing of the original text different from the final text. Second, the elimination of a requirement for exact spacing information in the original text (e.g., this field is 7 characters long) might also reduce maintenance, assuming that FDS can appropriately format the information as it produces the finished document. This feature, too, renders the spacing of the original text different from the final document.

Voice-Input Device Study

By agreement between SAI and MPAD, this issue was not explicitly investigated. We did become aware, however, that a similar study is currently underway at NASA Ames.